Spring 2024

Performing Holt-Winters Time Series Forecasting Using Neural Network Based Models

Kazeem Olanrewaju Bankole

PERFORMING HOLT-WINTERS TIME SERIES FORECASTING USING NEURAL NETWORK BASED MODELS

by

KAZEEM BANKOLE

(Under the Direction of Ionut E. Iacob)

ABSTRACT

We show how to create Artificial Neural Network based models for performing the well-known Holt-Winters time series analysis. Our work fares well compared to the well-known Holt-Winter time series prediction method while avoiding the burden of searching for the parameters of the model. We present the theoretical justification of the connection between the two models and experimental results showing the similarities of these models.

INDEX WORDS: Artificial Neural Networks, time series analysis, prediction

2009 Mathematics Subject Classification: 15A15, 41A10

PERFORMING HOLT-WINTERS TIME SERIES FORECASTING USING NEURAL NETWORK BASED MODELS

by

KAZEEM BANKOLE

B.Sc. University of Ilorin, Nigeria, 2018

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

PERFORMING HOLT-WINTERS TIME SERIES FORECASTING USING NEURAL NETWORK BASED MODELS

by

KAZEEM BANKOLE

Major Professor:	Ionut E. Iacob
Committee:	Felix Hamza-Lup
	Zheni Utic

Electronic Version Approved:
May 2024

DEDICATION

This thesis is dedicated to my dearest wife, Zainab Abdul-Rahman. I want to express my profound gratitude for your unwavering love, support, and understanding. Your belief in me has not only fueled my determination but has been a guiding light throughout this journey. Your presence has brought warmth and joy, making every step of this journey a cherished memory.

To my baby, Aryan Bankole, your sweet giggles and insatiable curiosity bring us immeasurable joy. You are not just a symbol of hope, but the very embodiment of a bright future. This thesis is a tribute to my aspirations for you and the world you will grow up in, a world I am striving to make better through my academic pursuits.

To my family, I owe my success. Your support gave me the strength to overcome challenges and pursue my dreams.

This thesis is dedicated to each of you with my profound gratitude and love. You are my inspiration, my motivation, and my greatest blessing.

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my thesis supervisor, Dr. Ionut E. Iacob, for his exceptional guidance, unwavering support, and constructive feedback while developing this thesis. Without his invaluable contributions, this research would not have been possible. Dr. Iacob's dedication and commitment to my academic growth have been instrumental in shaping my research, and I truly appreciate his efforts in helping me achieve my goals. I cannot thank him enough for his patience, encouragement, and expertise, which have undoubtedly enhanced the quality and impact of my work.

I also sincerely appreciate my committee members, Dr. Felix Hamza-Lup and Dr Zheni Utic, for their valuable feedback, constructive criticism, and encouragement. Your expertise and input have greatly enhanced the quality and depth of this work.

Finally, I thank all those who have contributed to my academic and personal growth through discussions, encouragement, or simply being there when needed. Your support has been invaluable, and I am deeply grateful for it.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

LIST OF SYMBOLS

$\mathbb{R}$	Real Numbers
$\mathbb{C}$	Real Numbers
$\mathbb{Z}$	Integers
$\mathbb{N}$	Natural Numbers
$\mathbb{N}_0$	Natural Numbers including 0
$L_p(\mathbb{R})$	p-integrable functions over $\mathbb{R}$
$L(X, Y)$	Linear maps from X to Y
$\operatorname{rank}(T)$	Rank of a linear map

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The analysis of time series data is integral in many fields, as it can reveal patterns, trends, and dependencies within sequential information. It is essential for making informed decisions, predicting future outcomes, and comprehending dynamic phenomena across various disciplines such as finance, economics, climate science, and healthcare. With its valuable insights, time series analysis enables accurate forecasting and informed decision-making in various fields.

Time series can be mathematically described as sequences of numbers, typically obtained from measurements at equal intervals of time (hence their name). They are ubiquitous in many types of processes (ranging from temperature measurements, energy consumption, stock prices, patients' heartbeats or brain signals, etc.), and it comes as no surprise that their study is of continuous research interest.

One of the very simple, effective, and popular methods for time series analysis is the exponential smoothing introduced by Brown [2], Holt [9], and Winters [23] in the late 1950s. They used weighted averages of past observations (with weights decaying exponentially while going back in time) to forecast future observations. The method was successfully used to perform time series analysis and forecasting for various types of time series.

In [13], the authors use the Holt-Winters method on a few time series from the UCI Machine Learning Repository [12]. The method is applied in [21] for predicting natural gas production, in [16] for forecasting rice production, and in [19] for analyzing abaca fiber data. The works in [20] and [17] present more applications of the Holt-Winters method.

The Holt-Winters method presents some long-known limitations [3], such as find-

ing good parameters or the fact that the parameters are chosen for the whole time series sequence, whereas different fragments of the time series data may present different characteristics. Additional optimization methods (such as the Nelder-Mead [18] method) or machine learning techniques are typically used to find the best parameters.

The Artificial Neural Networks (ANN) were built on early ideas of modeling human brain judgments [15] and emerged as very powerful machine learning models in the early 1990's. They were very good candidate models for time series analysis and forecasting from their very inception [8]. While currently producing excellent data analysis results, ANN based models are known to be complicated (hundreds or thousands of parameters), difficult to understand (in terms of how they model the data) models.

This research aims to establish a simple connection between the popular Holt-Winters and Neural Networks models. We show how to build Artificial Neural network models to perform the Holt-Winter type of time series analysis and subsequently extend such models to perform more accurate forecasting. Our methods combine the best of two worlds: the simplicity and robustness of the Holt-Winters method with the effectiveness and power of Neural Networks-based models.

The rest of this research is organized as follows. In Chapter 2, we give some background on the Times Series and its decompositions, Holt-Winters method, and Artificial Neural Network models. The proposed Neural Network based model for time series analysis is presented in Chapter 3. Chapter 4 presents the experimental results, and the conclusion and future work are presented in Chapter 5.

1.2 Related work

In the pursuit of advancing forecasting methodologies [14] conducted a study comparing artificial neural networks (ANNs) and traditional time series models for forecasting commodity prices. A feedforward neural network capable of modeling nonlinear relation-

ships was used to compare ARIMA and neural networks to analyze wheat and live cattle prices from 1950 to 1990. The experimental design involved seven iterations over successive three-year periods, utilizing a walk-forward or sliding window approach from 1970 to 1990, thereby generating out-of-sample results.

According to [14], the neural network models outperformed the autoregressive integrated moving average (ARIMA) model regarding forecasting accuracy. Notably, the neural network models achieved a notable 27% and 56% lower mean squared error than the ARIMA model. Additionally, assessments based on absolute mean error and mean absolute percent error consistently favored the neural network models.

The study found that neural network models can detect significant turning points for wheat and cattle prices, whereas the ARIMA model only predicted wheat prices. Furthermore, [14] emphasized the versatility of this forecasting method, noting its non-specificity to particular problems and reliance solely on past prices. This characteristic makes it applicable to diverse forecasting challenges, extending its utility beyond commodity prices to encompass domains such as stocks and other financial instruments.

Also, the studies of [22] compare ANNs against the Box-Jenkins methodology for forecasting and established that ANNs are comparatively better for forecasting problems with long forecasting horizons.

CHAPTER 2

BACKGROUND

Next, we present some necessary background on Holt-Winters models for time series smoothing and trending, as well as the Artificial Neural Network model. In the presentation, we use the notation $[x_k]$ to denote a time series (sequence of data points), $k \geq 1$.

2.1 TIME SERIES

A time series is a set of data collected at time points, observations, or measurements recorded at successive and equally spaced intervals over time.

Time series analysis involves exploring, modeling, and extracting meaningful insights from this chronological data (time order), making it a fundamental tool in various domains such as finance, economics, weather forecasting, and signal processing.

Mathematical representation:

$$Y_t = \underbrace{T_t}_{\text{Trend}} + \underbrace{S_t}_{\text{Seasonal}} + \underbrace{C_t}_{\text{Cyclical}} + \underbrace{\varepsilon_t}_{\text{Error}}$$

2.1.1 COMPONENTS OF TIME SERIES

1. Trend Variation (T_t): A consistent and sustained increase or decrease in the data over an extended period of time is referred to as a trend. Similarly, a Trend is the long-term systematic movement or trajectory in a time series that captures the overall direction, whether upward, downward, or stable.

Mathematical Representation:

$$y_t = T_t + \text{Seasonal}_t + \text{Cycle}_t + \varepsilon_t$$

where linear trend can be expressed as

$$T_t = \beta_0 + \beta_1 \cdot t$$

where β_0 is the intercept, β_1 is the slope, ε_t is random error term, and t is time.

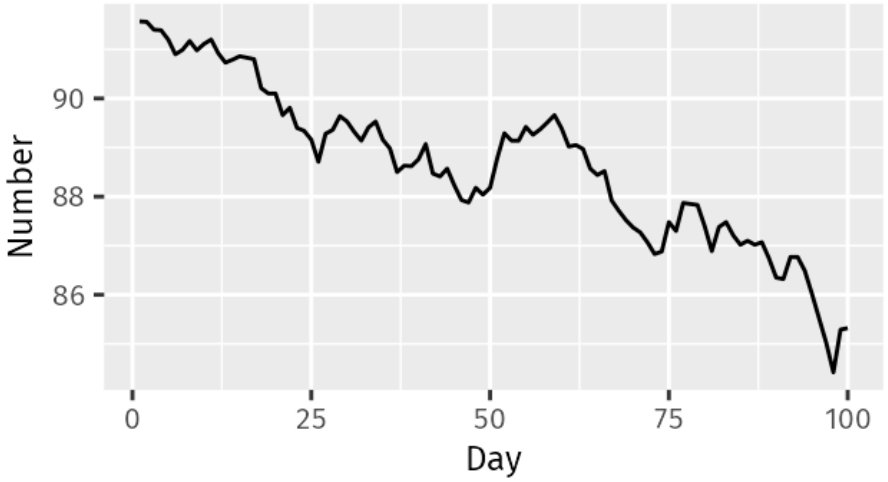

Figure 2.1: US Treasury bill contracts

Figure 2.1 shows the US Treasury Bill contracts results from the Chicago market over 100 consecutive trading days in 1981 [5]. Although there is no seasonality in the data, there is a clear and noticeable downward trend. If we had a much longer series of data, we could identify that this downward trend is part of a long cycle, but when we only consider a period of 100 days, it appears to be a simple trend.

2. Seasonality (S_t) : A seasonal variation or pattern exists when the series exhibits regular fluctuations during the same months, same quartile, or every year. Seasonality is always of a fixed and known period.

Mathematical Representation:

$$y_t = T_t + \mathrm{S}_t + \mathrm{C}_t + \varepsilon_t$$

A classic example is monthly temperature data for a location that experiences seasonal changes throughout the year, where S_t represents the seasonal effect.

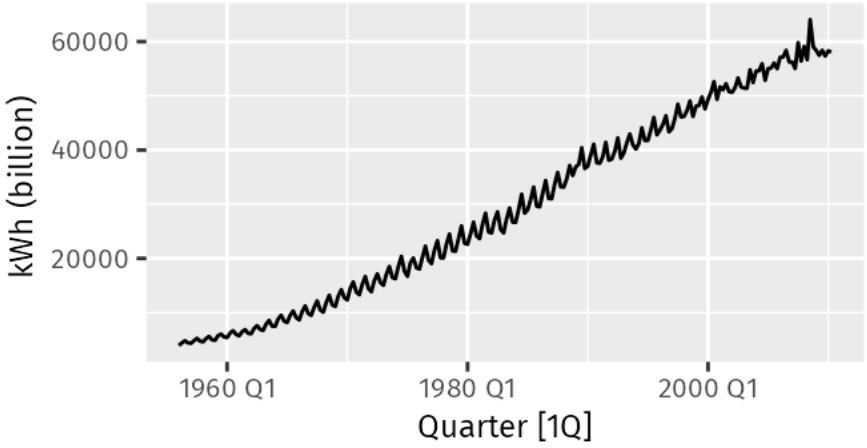

Figure 2.2: Australian Quarterly Electricity production

Figure 2.2 displays the quarterly electricity production in Australia with a strong increasing trend and seasonality [5]. There is no evidence of cyclic behavior.

3. Cyclicity (C_t): A cycle refers to a pattern in which data shows inconsistent increases and decreases. Economic conditions primarily cause these fluctuations, which are commonly associated with the ”business cycle.” These fluctuations usually last for at least two years.

If the fluctuations in a dataset do not follow a fixed frequency, they are considered cyclic. However, the pattern is considered seasonal if the frequency is consistent and related to some aspect of the calendar. On average, cycles last longer and have more variable magnitudes than seasonal patterns.

Figure 2.3 displays the monthly housing sales show strong cyclic behavior in a period

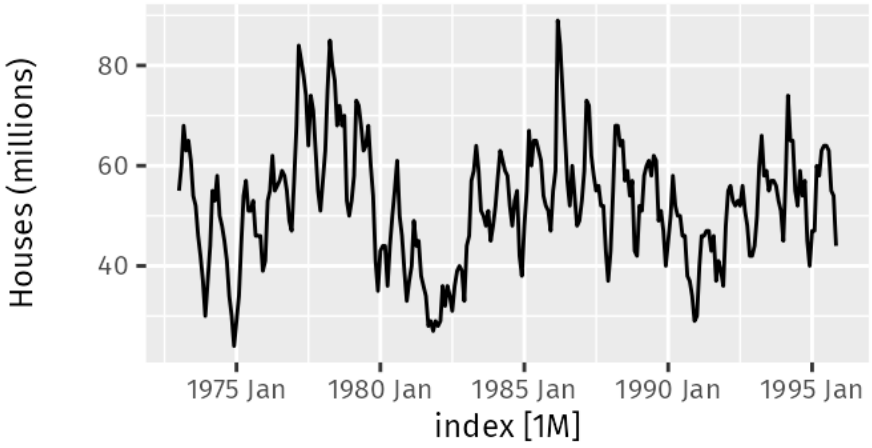

Figure 2.3: Sales of new one-family houses in USA

of about 6–10 years [5].

4. **Random or Residual Variation (ε_t):** The irregular or residual component, denoted as (ε_t), is the unexplained portion of a time series at a specific time t. It captures the random and unpredictable elements that are not accounted for by the identified systematic components of the time series.

   Figure 2.4 displays the daily changes in Google's closing stock price, do not exhibit any discernible trend, seasonality, or cyclic behavior [5]. They are random fluctuations that are not very predictable, and there are no strong patterns that could assist in developing a reliable forecasting model.

2.1.2 Types Of Time Series Decomposition

1. **Additive Decomposition:** In additive decomposition, a time series is expressed as the sum of its components.

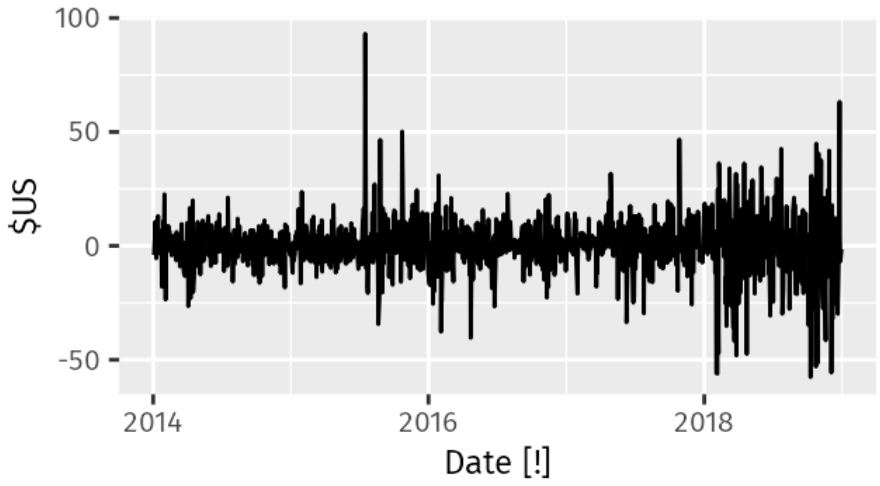

Figure 2.4: Daily Changes in Google Closing Stocks Prices

$$Y_t = T_t + S_t + C_t + \varepsilon_t$$

2. **Multiplicative Decomposition:** Represents the time series as the product of multiplicative components, allowing for interaction effects between components.

$$Y_t = T_t * S_t * C_t * \varepsilon_t$$

2.1.3 Measurements and metrics

1. **Measurements for Trend:**

   - Linear Regression Slope: Indicates the rate of change in the time series, representing the trend.

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

- Moving Averages: This smoothing out short-term fluctuations, revealing underlying trends over a specified period.

$$\text{Simple Moving Average (SMA)}: \quad \bar{Y}_t = \frac{1}{k}\sum_{i=1}^{k} Y_{t-i+1} \tag{2.1}$$

$$\text{Exponential Moving Average (EMA)}: \quad EMA_t = \alpha \cdot Y_t + (1-\alpha) \cdot EMA_{t-1} \tag{2.2}$$

  - $\bar{Y}_t$ represents the simple moving average of Y at time t.
  - k is the number of periods used for the calculation.
  - Y_{t-i+1} represents the value of Y at time $t-i+1$.
  - EMA_t is the exponential moving average at time t.
  - α is the smoothing factor, a parameter between 0 and 1.

- Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF): ACF and PACF help identify trends by showing the correlation between a time series and its lagged values.

2. **Measurements for Seasonality:**

- Seasonal Indices S_i: Seasonal indices represent the relative strength of seasonality at different points in the year, where $\bar{Y}_i$ is the average for season i and $\bar{Y}$ is the overall average.

$$S_i = \frac{\bar{Y}_i}{\bar{Y}}$$

- Fourier Transform: Fourier Transform analyzes the frequency components of time series data, useful for identifying seasonality.

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i f t} dt$$

3. **Irregular Measurement:** Residuals ε_t are the differences between the observed values (Y_t) and the predicted values of the trend ($\hat{T}_t$) and seasonality ($\hat{S}_t$).

$$\varepsilon_t = Y_t - (\hat{T}_t + \hat{S}_t)$$

2.2 FORECASTER'S TOOLBOX

The process of forecasting time series data involves several steps [10].

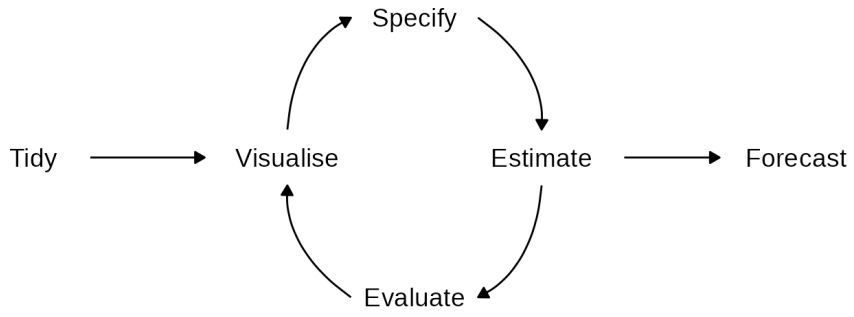

Figure 2.5: Prediction Workflow

1. Data Preparation (Tidy): The initial step in forecasting involves data preparation, including loading, identifying missing values, filtering time series, and performing pre-processing tasks [6].

2. Visualise the Data (Plot): Understanding data requires visualizing it to detect common patterns, allowing you to specify an appropriate model.

3. Define a Model (Specify): Choosing the appropriate time series model is crucial for generating accurate forecasts. Section 2.4 provides an in-depth discussion of models.

4. Train the Model (Estimate): After determining an appropriate model, the next step is to train the model using some data. Section 2.7 speaks more about the steps involved in training a model.

5. Check Model Performance (Evaluate): After fitting a model, evaluating its performance on the data is crucial. Various diagnostic tools are available that can help understand the model's behavior. Additionally, accuracy measures enable comparison of different models against each other.

6. Produce Forecasts (Forecast): It is time to produce the forecasts after specifying an appropriate model, estimating it, and checking it. The easiest way to produce the forecasts is by specifying the number of future observations to forecast.

2.3 THE HOLT-WINTERS MODELS FOR TIME SERIES ANALYSIS

In the Holt-Winter model, a time series $[x_k]$ is used to compute an exponentially weighted moving average (EWMA) sequence $[s_k]$, which is a smoothed version of the original time series:

$$s_{k+1} = \alpha x_k + (1-\alpha)s_k, \;\; k \geq 1, \;\; \alpha \in (0,1) \tag{2.3}$$

Where α is a parameter of the model. Clearly, the smoothed version of the model is recursively (repeatedly) constructed as a weighted average between the previous data point and smoothed values. One can see s_{k+1} as the future (predicted) value, computed using the last point data and predicted values. The chosen value of parameter α balances between the importance of the original value and the predicted value, with $\alpha > 0.5$ giving more weight to the actual data point value (less smoothing) and $\alpha < 0.5$ giving more weight to the predicted value (more smoothing).

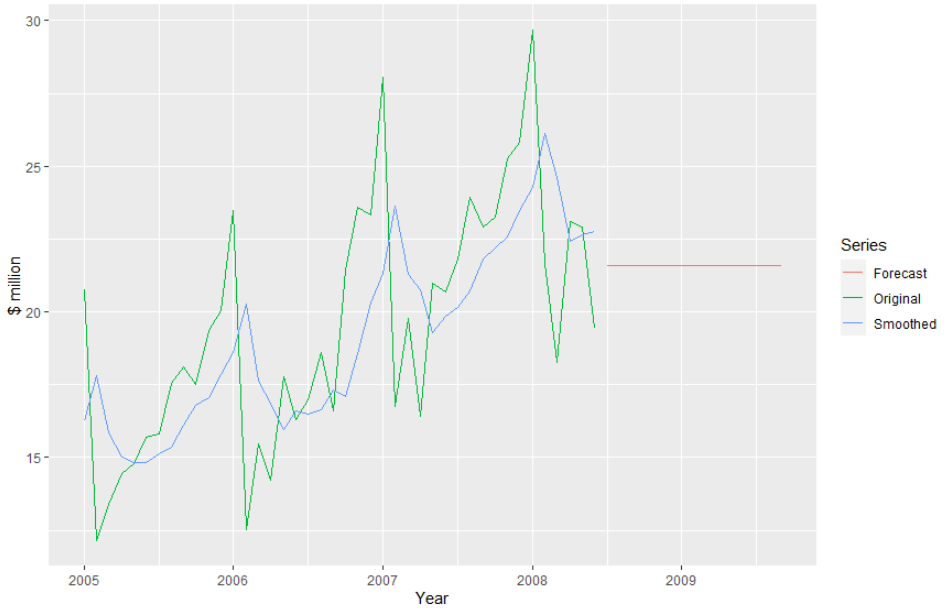

Figure 2.6: Anti-diabetic drug sales and forecasting using EWMA (2.3)

Figure 4.1 shows how the smoothing method performs on "Anti-diabetic drug sales" data [11] and the future sales forecasting using (2.3). The figure illustrates a clear inability of the model to capture future trends (increasing or decreasing) for future values. To overcome this limitation, the model (2.3) was improved by Holt [9] to include a trend component, as follows.

$$s_{k+1} = \alpha x_k + (1-\alpha)(s_k + y_k) \tag{2.4}$$

$$y_{k+1} = \beta(s_{k+1} - s_k) + (1-\beta)y_k \tag{2.5}$$

$$y_{k+f} = s_k + f y_k \tag{2.6}$$

where s_k is the smoothed version, y_k the trend (raise) and y_{k+f} is the prediction of f steps in the future. As (2.4) shows, the smoothed series contains the contribution of the trend factor (2.5), which is a weighted average of the smoothed raise and the previously estimated raise (trend). The model now contains two parameters, α and β, which represent the weights for computing the smoothed and trend values, respectively. Finding good values for these pa-

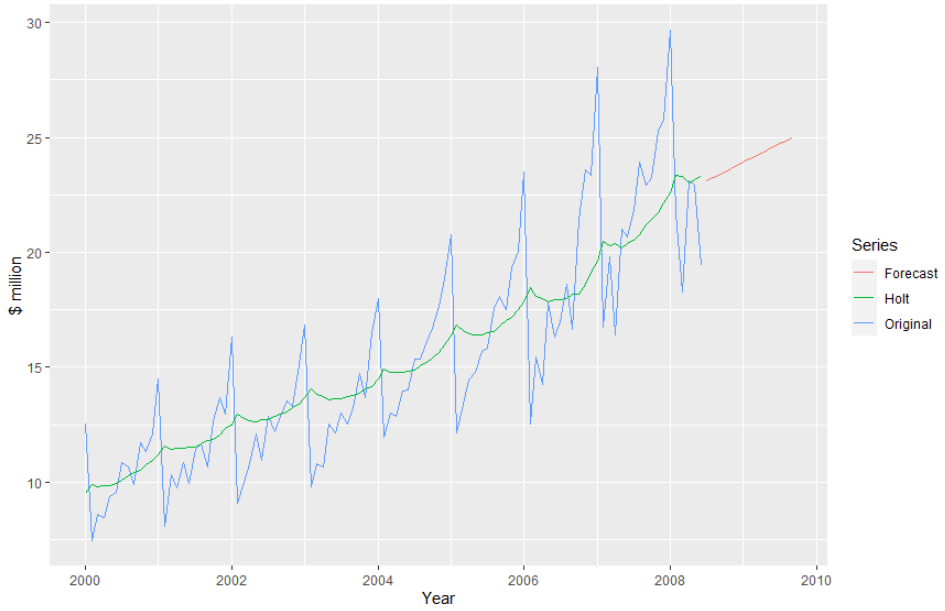

Figure 2.7: Anti-diabetic drug sales and forecasting using Holt's trending model (2.6)

rameters is a practical challenge, which is often times overcome using various optimization and/or machine learning methods. Figure 4.2 shows the improvement in forecasting the "Anti-diabetic drug sales" when (2.6) is used. Additionally, Holt method (2.4) performs a "double exponentiation smoothing", which results in a smoother series compared to (2.3) shown in Figure 4.1.

2.4 The Artificial Neural Network model

An Artificial Neural Network (ANN), depicted in Fig. 2.8, is a directed graph that performs a mapping $n : \mathbb{R}^n \to \mathbb{R}^m$ from the input space $\mathbb{R}^n$ to the output space $\mathbb{R}^m$. An ANN consists of an input layer (or n nodes), any number of hidden layers (each with any number of nodes), and an output layer (with m nodes). Each layer node is connected to each node of the subsequent layer. Each edge of the ANN model has a weight associated with it and the $\vec{y} = \vec{y}(\vec{x})$ formula is computed progressively at each node, as one progresses from left

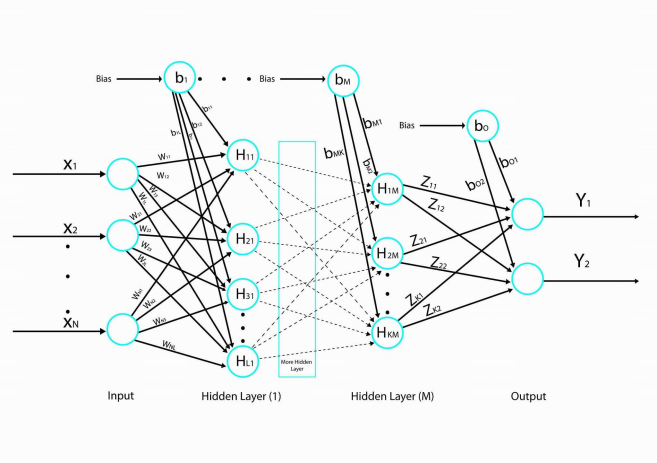

Figure 2.8: A generic ANN model

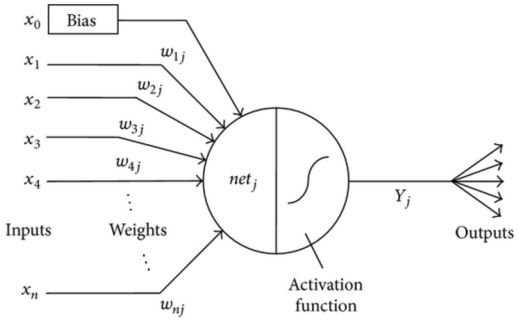

Figure 2.9: The computation detail at one ANN node level

(inputs) to the right (outputs) as follows. Every node j in ANN computes a real-valued output y_j through an integration (summation) function Σ of all the weighted outputs from preceding nodes, followed by an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ that normalizes the node output (for instance, the sigmoid function was initially used as an activation function). The process is depicted in Fig. 2.9. In summary, the output y_j of node j in terms of its inputs $\vec{x}_j$ is computed as:

$$y_j = \sigma\left(W_j \vec{x}_j\right)$$

Where W_j is the matrix of all weights for edges going into node j. When all nodes contributions are aggregated in the output layer, the network model computes the output as a

composition of the above:

$$\vec{y} = \sigma\left(W_h \sigma\left(W_{H-1} \sigma(\dots)\right)\right) \tag{2.7}$$

where H is the number of hidden layers and each W_i is the corresponding weights matrix for layer i.

2.5 Training a Time Series Forecasting Model

Step 1: Import Libraries

- Import necessary libraries such as NumPy for numerical operations, pandas for handling time series data, Keras for building the neural network, scikit-learn for preprocessing, and statsmodels for the Holt-Winters method.

Step 2: Load or Generate Time Series Data

- Load your time series data into a pandas DataFrame. This could be real-world data or generated data for experimentation.

Step 3: Train-Test Split

- When building a model, splitting your dataset into training and testing sets is essential. You should reserve a portion of your data specifically to test the model's performance.

Step 4: Preprocess and Normalize Data

- Normalize the data using Min-Max scaling or another appropriate scaling method. Normalization ensures that all input features are on a similar scale, which helps the model during training.

Step 5: Create Sequences for RNN

- Create sequences of input data for training the RNN. Each sequence includes a specified number of time steps (lags) to capture temporal dependencies in the data.

Step 6: Design and Train the RNN Model

- Design the architecture of the Recurrent Neural Network (RNN). Common choices include Long Short-Term Memory (LSTM) layers. Train the model using the training data, specifying the number of epochs (iterations) and batch size.

Step 7: Make Predictions and Evaluate

- Use the trained RNN model to make predictions on the test set. Evaluate the model's performance using metrics like Mean Squared Error (MSE). Visualize the predicted values against the true values to get insights into how well the model captures patterns in the data.

These steps provide a high-level overview of training an RNN for time series forecasting using the Holt-Winters method. Adjustments and refinements can be made to your dataset and forecasting outcomes based on their characteristics.

CHAPTER 3

TIME SERIES PROCESSING USING ANN MODELS

3.1 NEURAL NETWORK BASED MODELS FOR TIME SERIES SMOOTHING

We show that an ANN model (2.7) can be adapted to compute time series smoothing (2.3) and (2.4) by using the appropriate architecture and parameters. The first key observation is that by recursively applying the formula in (2.3) one would obtain:

$$\begin{aligned} rcls_{k+1} &= \alpha x_k + (1-\alpha)(\alpha x_{k-1} + (1-\alpha)s_{k-1}) = \cdots \\ &= \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} x_i + (1-\alpha)^k s_1 \end{aligned} \tag{3.1}$$

which, for the next predicted value, represents a weighted average of all past series values (except for the last term, which represents the first predicted value). The second key observation is that, by choosing $\sigma(x) = x$ (the identity function), formula (2.7) becomes $\vec{y} = W\vec{x}$, a linear combination of inputs where W is the product of all parameter matrices, for all hidden and output layers. Consequently, a model with a single hidden layer of a single node can produce such a linear combination of the inputs (for a single output, which would represent the next estimated value). Such a model, with weights chosen as the coefficients of x_i in (3.1), would therefore compute (2.3). There is one problem with this ANN model: it will produce a limited linear combination of n series values in the past (where n represents the number of input nodes), whereas (3.1) represents the linear combination of *all* values in the past. However, in practice, as the coefficients in (3.1) decay exponentially, the influence of series values in the distant past will fade quickly. For instance, by taking only the first 10 terms (corresponding to x_k, x_{k-1}, ..., x_{k-9}) in (3.1), the terms (past value contributions) that will be ignored will have the coefficients $\alpha(1-\alpha)^{10}$, $\alpha(1-\alpha)^{11}$, etc. of orders $O(10^{-11})$, $O(10^{-12})$, respectively. An ANN model with n inputs will consequently produce an order $O(10^{n+1})$ approximation for (3.1) (and, implicitly, for (2.3)). In summary, we use an ANN model with n inputs, one hidden layer with a single node, and one output

(as in Fig. 2.9 with no bias node) to approximate the next series value (2.3) as:

$$s_{k+1} \approx y = W\vec{x} \tag{3.2}$$

where $W = [w_1\ w_2\ \cdots\ w_n]$, $w_i = \alpha(1-\alpha)^{i-1}$, and $\vec{x} = [x_k\ x_{k-1}\ \cdots\ x_{k-n+1}]^T$.

Similarly to creating the above ANN model (3.2) for (2.3), we start applying (2.4) recursively to obtain:

$$\begin{aligned} s_{k+1} &= \alpha x_k + (1-\alpha)(s_k + y_k) \\ &= s_k + f y_k \end{aligned} \tag{3.3}$$

Then, similar to the model we created for (2.3), we use (3.3) to create an ANN model with n inputs, one hidden layer with a single node, and one output to approximate the next estimated value (2.4) as follows.

Definition 1. [ANN smoothing model] The ANN smoothing model consists in an ANN with n input nodes, one hidden layer with parameters W and a single hidden node, and one output node that computes the model output as:

$$s_{k+1} \approx y = W\vec{x} \tag{3.4}$$

where $W = [w_1\ w_2\ \cdots\ w_n]$, $w_i = \alpha(1-\alpha)^{i-1}$, and $\vec{x} = [x_k\ x_{k-1}\ \cdots\ x_{k-n+1}]^T$.

We call (3.2) and (3.4) the ANN models for performing time series smoothing and forecasting, respectively.

3.2 Neural Network based model for Holt trend formula

We discuss next how the trend model in (2.4)-(2.6) can be implemented using the ANN-based models. We first notice that the level equation (2.4) can be recursively applied to produce the level as a weighted average of the past signal values and predicted levels.

$$s_{k+1} = \alpha \sum_{i=1}^{k} (1-\alpha)^{k-i} x_i + \sum_{i=1}^{k} (1-\alpha)^{k-i} y_i \tag{3.5}$$

Next, the recursive equation (2.5) computes the trend factor y_{k+1} as a weighted average of level s_k differences, which is essentially a weighted average of the predicted level (discrete) derivative sequence.

$$y_{k+1} = \beta \sum_{i=1}^{k} (1-\beta)^{k-i}(s_{i+1} - s_i) + (1-\beta)^k y_1 \tag{3.6}$$

Finally, from (2.4), the discrete predicted levels differences $s_{i+1} - s_i$ are produced as a weighted average of the signal discrete differences.

$$\begin{aligned} s_{k+1} - s_k &= \alpha \sum_{i=1}^{k} (1-\alpha)^{k-i}(x_{i+1} - x_i) \\ &\quad + (1-\alpha)(y_k - y_{k-1}) \end{aligned} \tag{3.7}$$

Continuing the process, (3.6) gives us the trend differences as weighted average of second order level differences (second derivative).

$$\begin{aligned} y_k - y_{k-1} &= \beta \sum_{i=1}^{k} (1-\beta)^{k-i}(s_i - s_{i-2}) \\ &\quad + (1-\beta)^{k-1} y_1 - (1-\beta)^{k-2} y_1 \end{aligned} \tag{3.8}$$

Putting together all (3.5)-(3.8) we obtain the predicted level as a weighted average of first order differences, second order differences, etc.

$$\begin{aligned} s_{k+1} &= \alpha \sum_{i=1}^{k} (1-\alpha)^{k-i} x_i \\ &\quad + \sum_{i=1}^{k} \alpha^{p_i} \beta^{q_i} (1-\alpha)^{r_i} (1-\beta)^{t_i} (x_{i+1} - x_i) \\ &\quad + O(\alpha^{p_i} \beta^{q_i} (1-\alpha)^{r_i} (1-\beta)^{t_i}) \end{aligned} \tag{3.9}$$

where p_i, q_i, r_i, t_i are integers values that grow larger and larger as the process continues to take into account higher and higher order differences.

The ANN trend model with finite difference up to order O is created using an ANN model with n inputs, one hidden layer with a single node, and one output to approximate the next estimated value (3.10) as follows.

Definition 2. [ANN trend model] The ANN trend model consists in an ANN with n input nodes, one hidden layer with trainable parameters W and a single hidden node, and one output node that computes the model output as:

$$s_{k+1} \approx y = W\vec{x} \tag{3.10}$$

where $W = [w_1\ w_2\ \cdots\ w_n]$,
$\vec{x} = [x_k\ x_{k-1}\ \cdots\ x_{k-n+1}\ \Delta_k^{(1)}\ \cdots \Delta_{k-n+2}^{(1)}\ \Delta_k^{(2)}\ \cdots \Delta_{k-n+3}^{(2)}\ \Delta_k^{(3)}\ \cdots \Delta_{k-n+4}^{(3)}]^T$,
and $\Delta_j^{(o)} = x_j - x_{j-o}$ (with all discrete differences up to order O).

We implemented our ANN based models (smoothing and trend) and produced experimental results and comparisons with Holt models. The experimental results are presented in the subsequent chapter.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented our model in Python using Keras [4] library for implementing Neural Networks models. Our experiments were organized in two categories: (i) ANN-based models with pre-set parameters and comparisons with Holt-Winter methods, and (ii) optimized ANN-based models parameters. For each experiment, we have used both artificial and real data. We performed our experiments on a PC Desktop computer using an Intel(R) Core(TM) i7-10700K CPU 3.80GHz processor equipped with 16GB of RAM. The experimental results are presented in the subsequent sections.

4.1 MODEL IMPLEMENTATION

The Python implementations of the EWMA and Hold models are presented in Listings 4.1.

```
def ewma(x, alpha: float) -> float:
    if len(x) < 2:
        return x[0]

    return (alpha * x[-1]) + ((1 - alpha) * ewma(x[:-1], alpha))

def holt(x, alpha: float,
            beta: float, initial_trend: float) -> float:
    if len(x) < 2:
        return initial_trend

    s = ewma(x, alpha)
    s0 = ewma(x[:-1], alpha)

    y0 = holt(x[:-1], alpha, beta, initial_trend)

```

```
    return (beta * (s - s0)) + ((1 - beta) * y0)
```

Listing 4.1: The EWMA and Holt models implementation

```
def identity_activation(x):
    return x

def ann_model(x_train):
    model = keras.Sequential()
    model.add(Dense(1, activation=identity_activation, input_shape=(
    x_train.shape[1], )))
    #model.add(Dense(1, activation="relu", input_shape=(x_train.shape
    [1], )))
    model.compile(optimizer='adam', loss='mean_squared_error')
    #model.summary()
    return model

def train(model, x_train, y_train, batch = 4, epochs = 500):
    history = model.fit(x_train, y_train, batch_size= batch, epochs=
    epochs)
    return history


def ann_forecast(model, x_train, datachunk = DATACHUNK, future = future)
    :
    x =  np.reshape(x_train[-1], (1, x_train.shape[1]))
    y = model.predict(x)
    x = np.concatenate((x[:,1:datachunk],y),axis = 1)
    f = []
    f.append(y[0])
    for i in range(future):
        y = model.predict(x)
        x = np.concatenate((x[:,1:datachunk],y),axis = 1)
```

```
        f.append(y[0])
    f = np.array(f)
    #f = f.reshape(-1)
    return f


def ann_trendforecast(model, x_train, datachunk = DATACHUNK, order = 3, future = future):
    dord = min(order, datachunk-1)
    x =  np.reshape(x_train[-1], (1, x_train.shape[1]))
    y = model.predict(x)
    x = np.concatenate((x[:,1:datachunk],y),axis = 1)
    f = []
    f.append(y[0])
    for i in range(future):
        for k in range(dord):
            x = np.concatenate((x, x[:,(k+1):(datachunk)]-x[:,(0):(datachunk-k-1)]), axis = 1)

        x =  np.reshape(x, (1, x_train.shape[1]))
        y = model.predict(x)
        x = np.concatenate((x[:,1:datachunk],y),axis = 1)
        f.append(y[0])
    f = np.array(f)
    #f = f.reshape(-1)
    return f
```

Listing 4.2: The ANN based model implementation

4.2 Experimental results

We performed two types of experiments. Firstly, we used ANN models as defined in (3.2) and (3.4) to perform forecasting (no ANN model training, we use pre-computed parameters) and subsequently compared the results with Holt-Winters forecasting. Secondly, we used the same ANN models architecture but the parameters were optimized to fit the original time series data (we train the model).

Figures 4.1 and 4.2 show the time series processing results on the anti-diabetic drug sales dataset [1] using the EWMA and Holt methods, respectively.

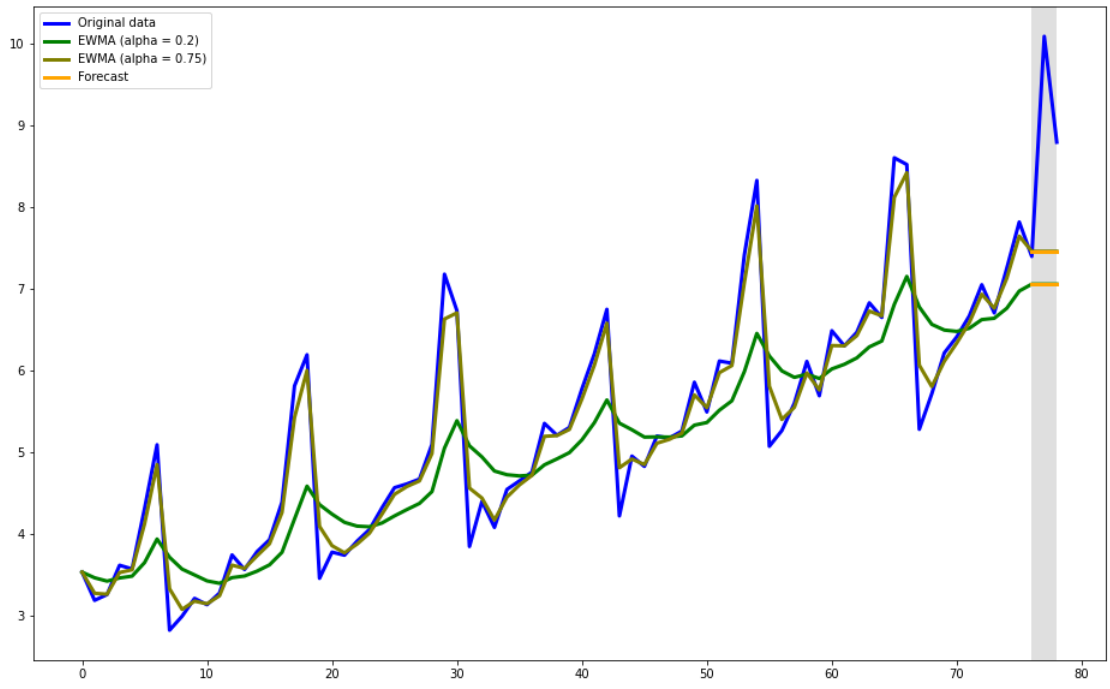

Figure 4.1: Anti-diabetic drug sales and forecasting using EWMA (2.3)

We subsequently performed two sets of experiments on the Anti-diabetic drug sales dataset [1] with our ANN-based smoothing model: using 5 and 10 past values, respectively (history 5 and 10, respectively).

Figure 4.3 and Figure 4.4 show the training and time series analysis for the ANN

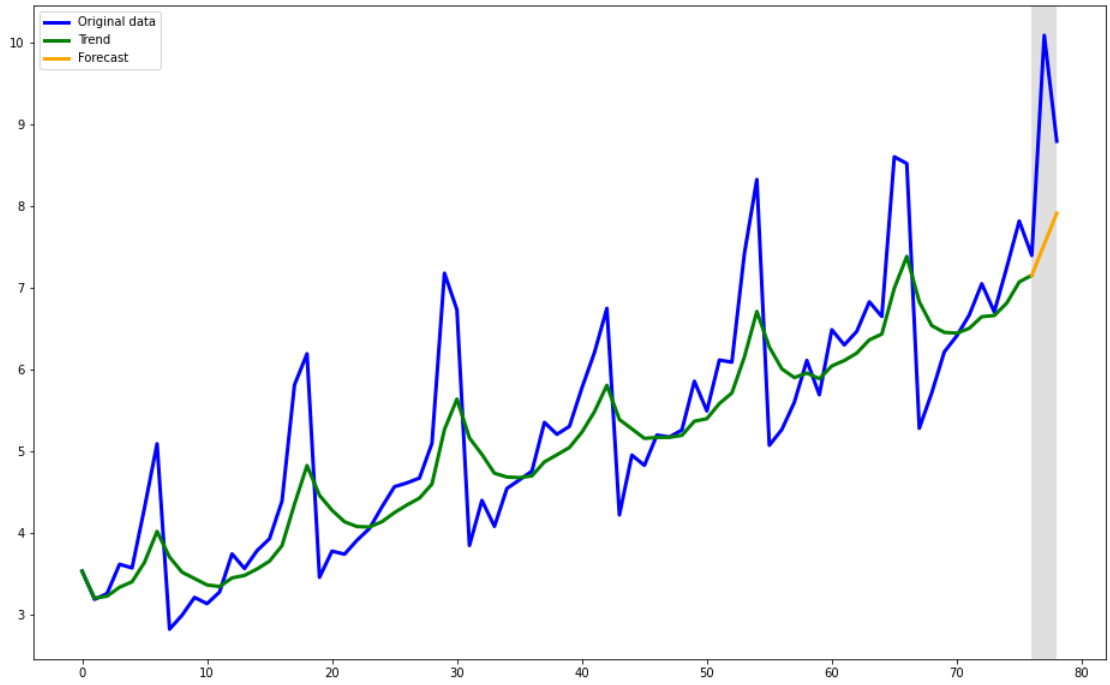

Figure 4.2: Anti-diabetic drug sales and forecasting using Holt's trending model (2.6)

smoothing model with history 5, respectively. The SSE of the model analysis in Figure 4.4 is 0.01781823370352002.

Figure 4.6 and Figure 4.7 show the training and time series analysis for the ANN smoothing model with history 10, respectively. The SSE of the model analysis in Figure 4.7 is 0.02613138994938639.

It is worth noting that the model with history 10 (10 past values being considered) is trained significantly faster than the model with history 5 (as shown in Figure 4.3 and Figure 4.6). However, as Figure 4.5 and Figure 4.8 show, both models' prediction is mostly influenced only the past term in history. This is consistent with Holt's model where the contribution of past terms fades quickly (exponentially).

We subsequently performed two sets of experiments on the Anti-diabetic drug sales dataset [1] with our ANN-based trend model: using 3 and 5 past values, respectively (history 3 and 5, respectively), with discrete difference order up to 3. The results are presented

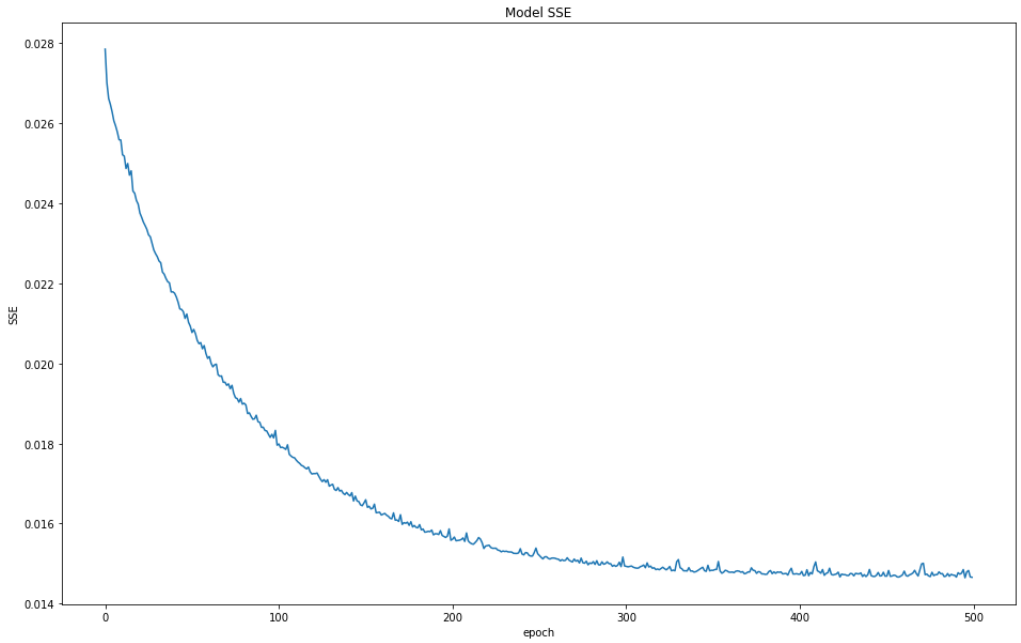

Figure 4.3: ANN smoothing model training (history = 5)l

in Figures 4.9-4.14.

The last set of experimental results led us to the following conclusions.

- The training of trend models is significantly faster, as illustrated by Figure 4.9 and Figure 4.12.

- Better time series smoothing (smaller SSE) is achieved with significantly smaller contributions from the past terms. We attribute this performance increase to taking into account not only past terms values, but also the discrete derivatives (differences) of recent past terms. In some sense, our ANN trend model simulates a Taylor series approximation, which is something we consider interesting for further investigation.

- Figure 4.11 and Figure 4.14 clearly show the significant contribution of recent past terms as well as their differences in predicting subsequent values. As a matter of fact, the contribution of differences seems to be more important than the contribution of the past terms. Moreover, the contributions of differences of order 1, 2, and 3 are all

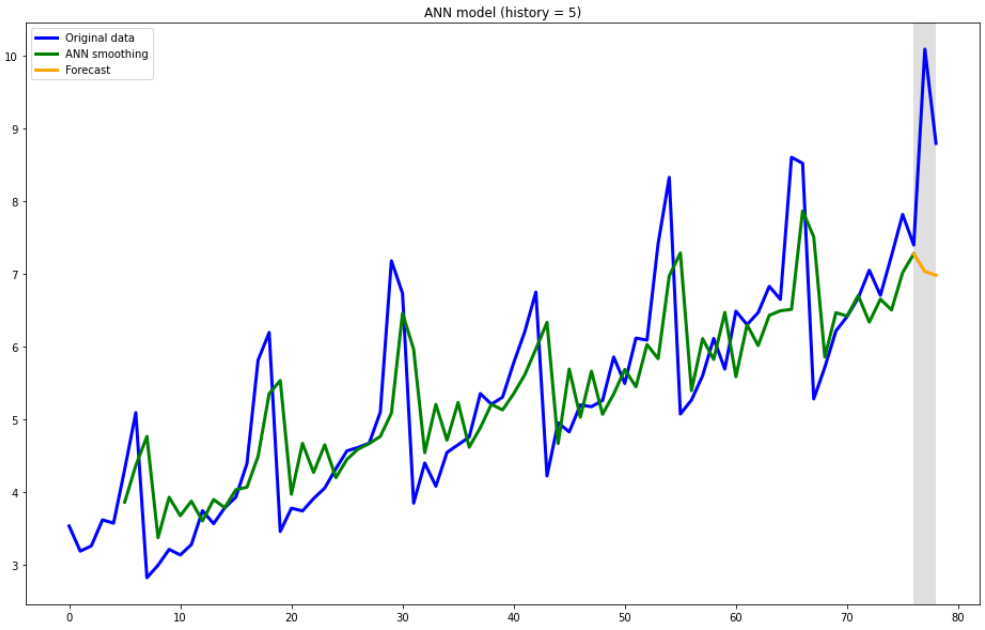

Figure 4.4: Anti-diabetic drug sales and forecasting using the ANN smoothing model (history = 5)l

important, but mostly from the most recent ones.

- Finally, the last set of experiments also shows the superiority of the ANN trend models for predicting future values.

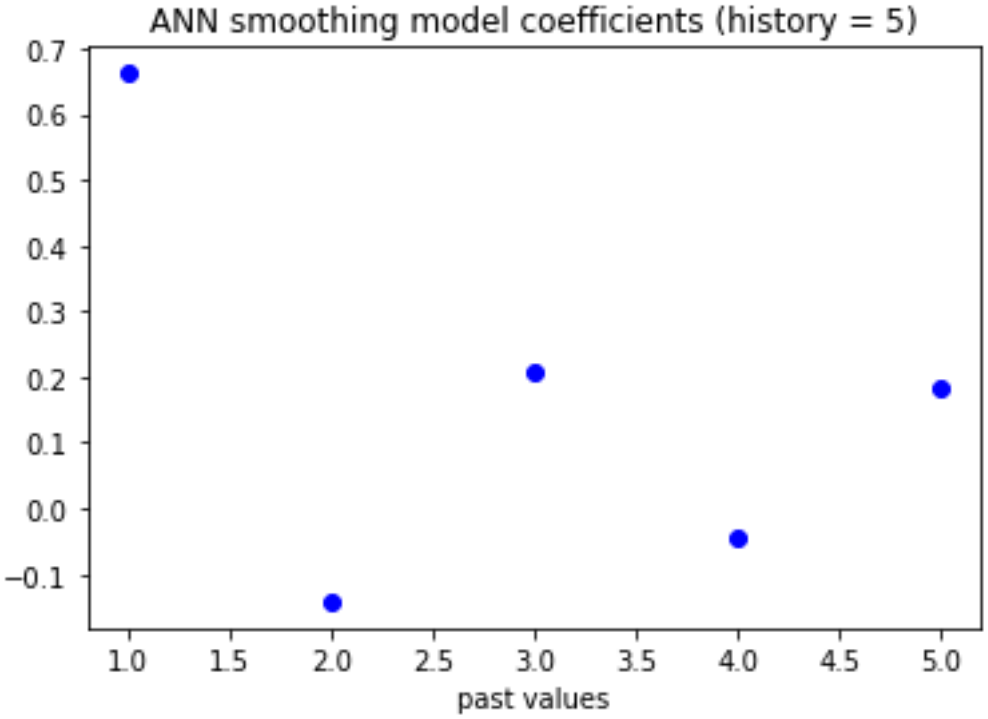

Figure 4.5: ANN smoothing model coefficients (history = 5)l

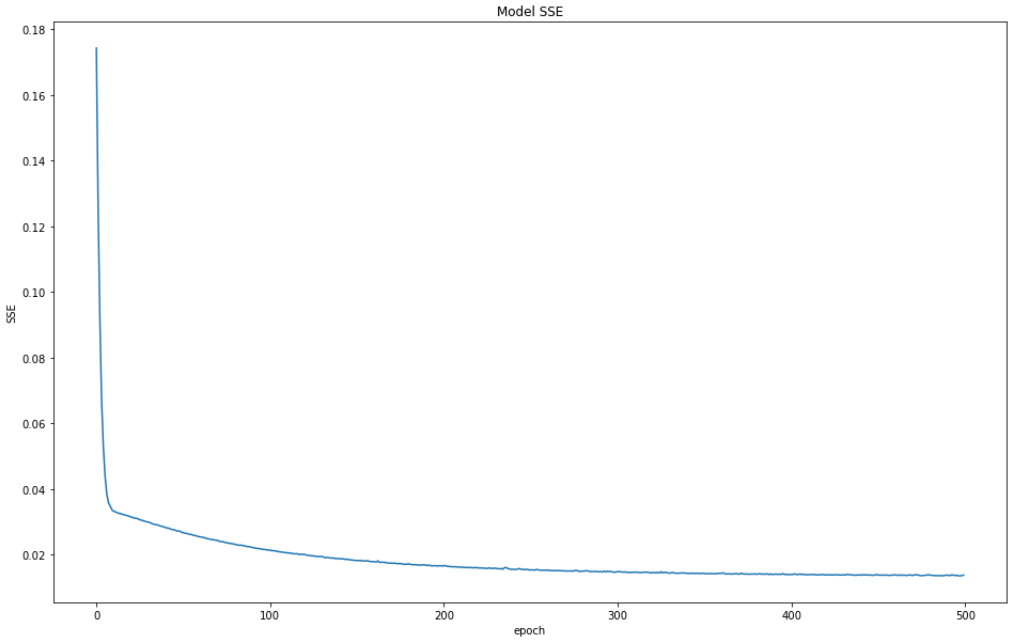

Figure 4.6: ANN smoothing model training (history = 10)l

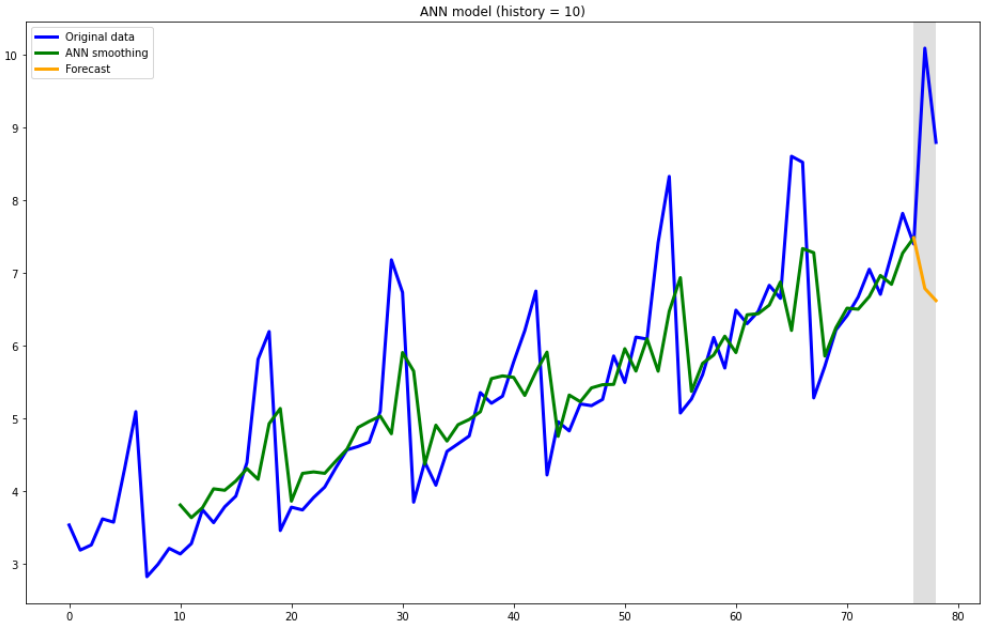

Figure 4.7: Anti-diabetic drug sales and forecasting using the ANN smoothing model (history = 10)l

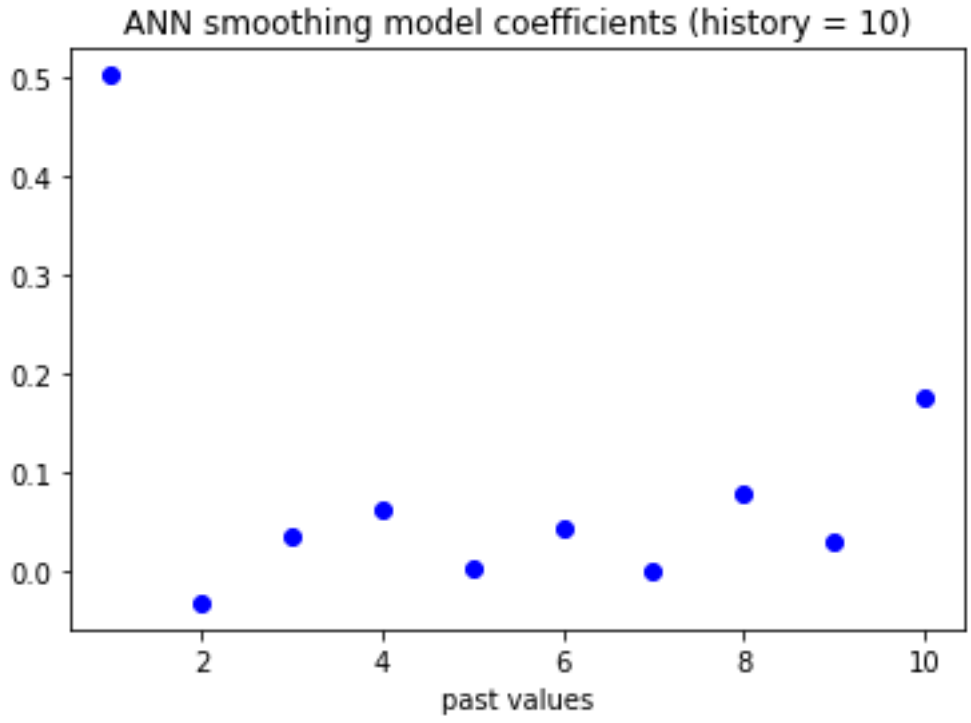

Figure 4.8: ANN smoothing model coefficients (history = 10)l

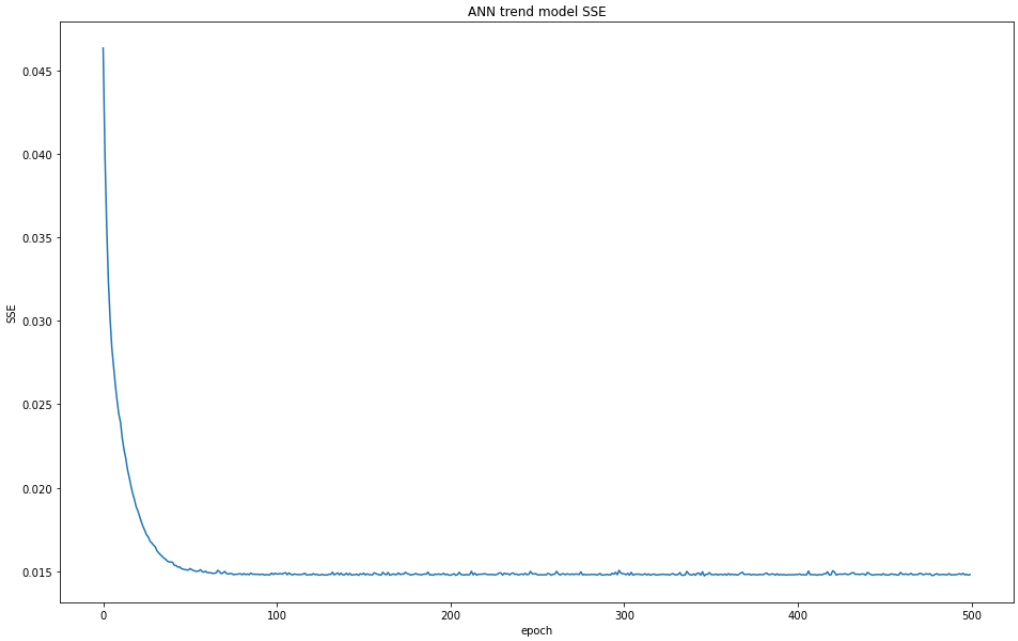

Figure 4.9: ANN trend model training (history = 3)

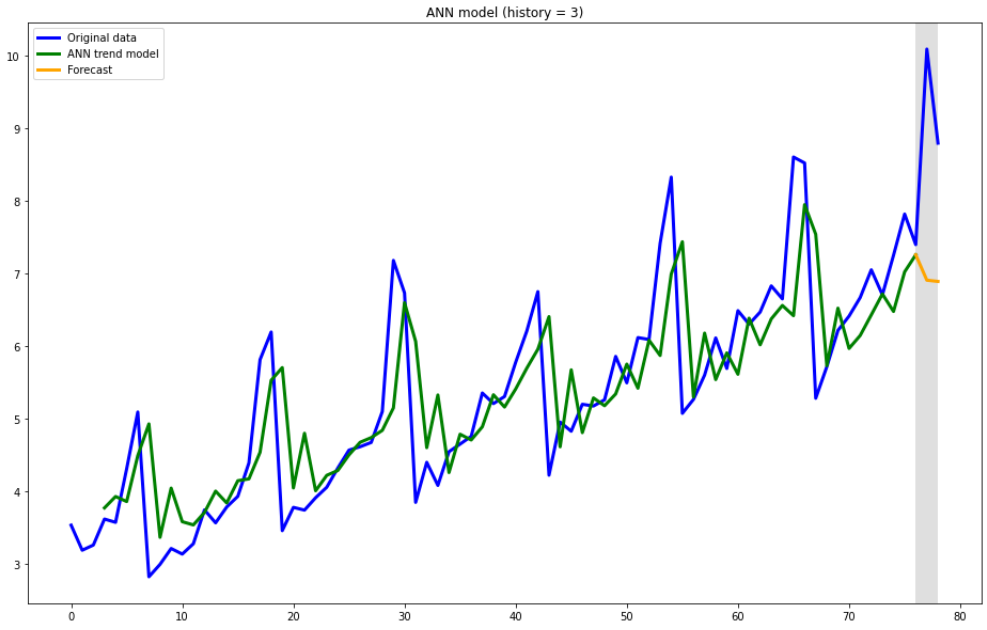

Figure 4.10: Anti-diabetic drug sales and forecasting using the ANN trend model (history = 3)

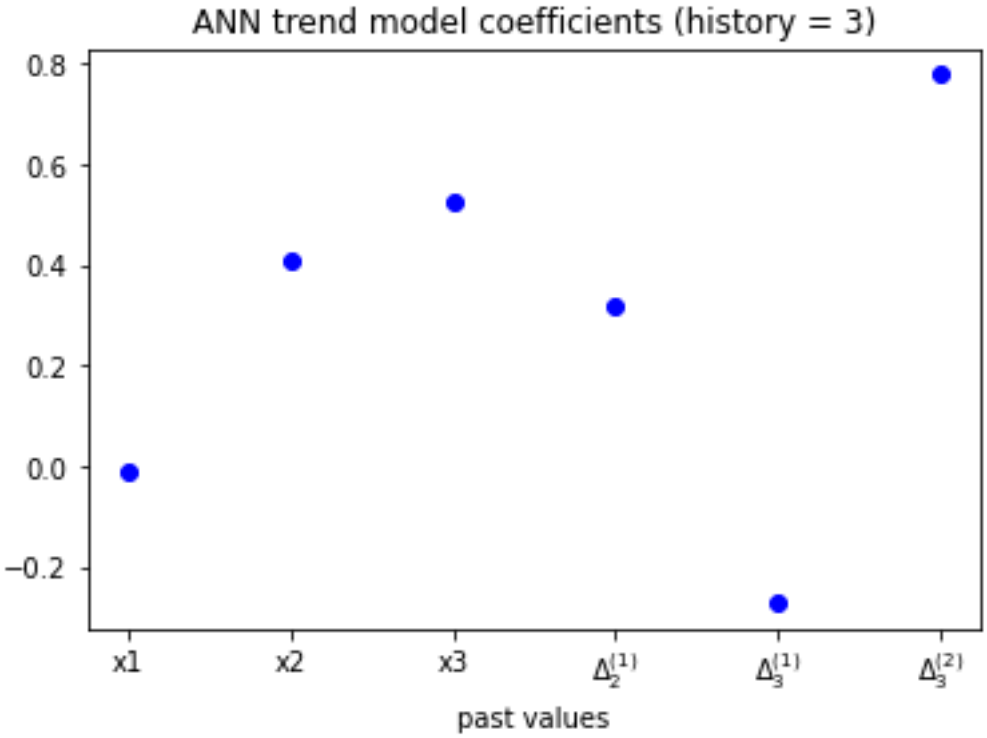

Figure 4.11: ANN trend model coefficients (history = 3)

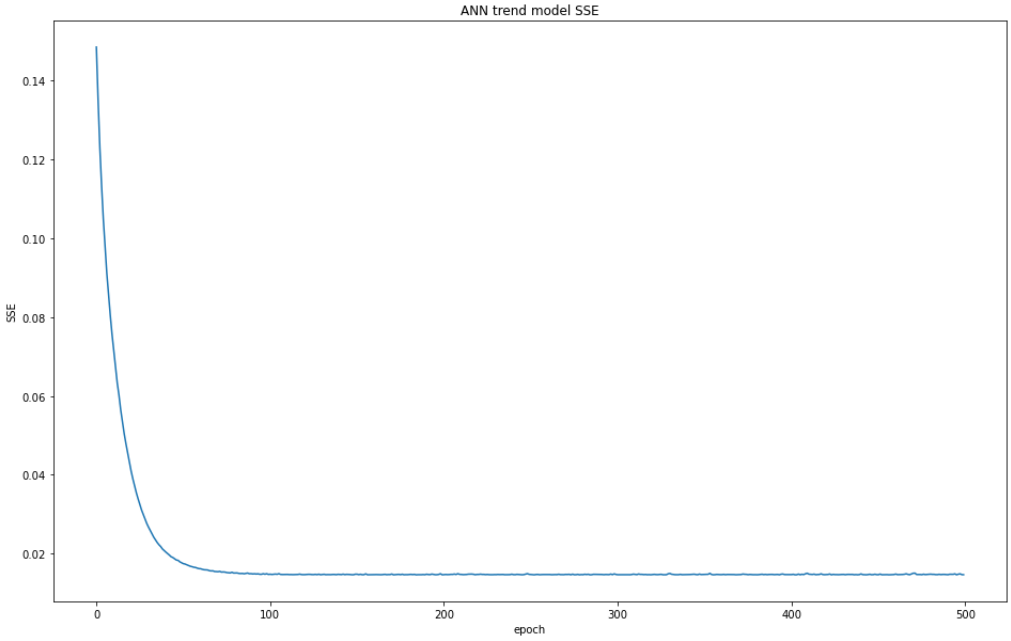

Figure 4.12: ANN trend model training (history = 5)

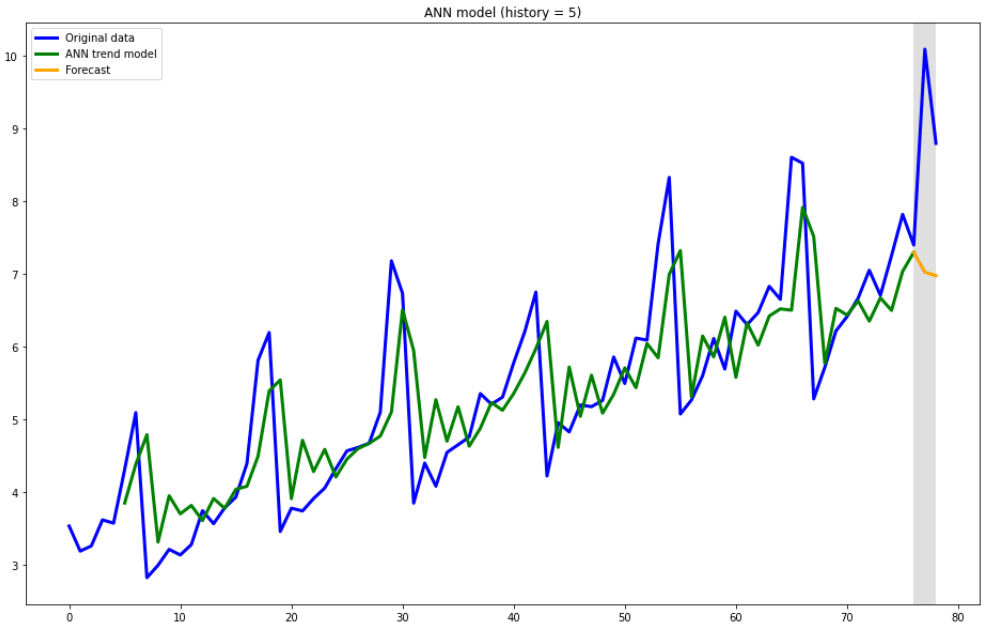

Figure 4.13: Anti-diabetic drug sales and forecasting using the ANN trend model (history = 5)

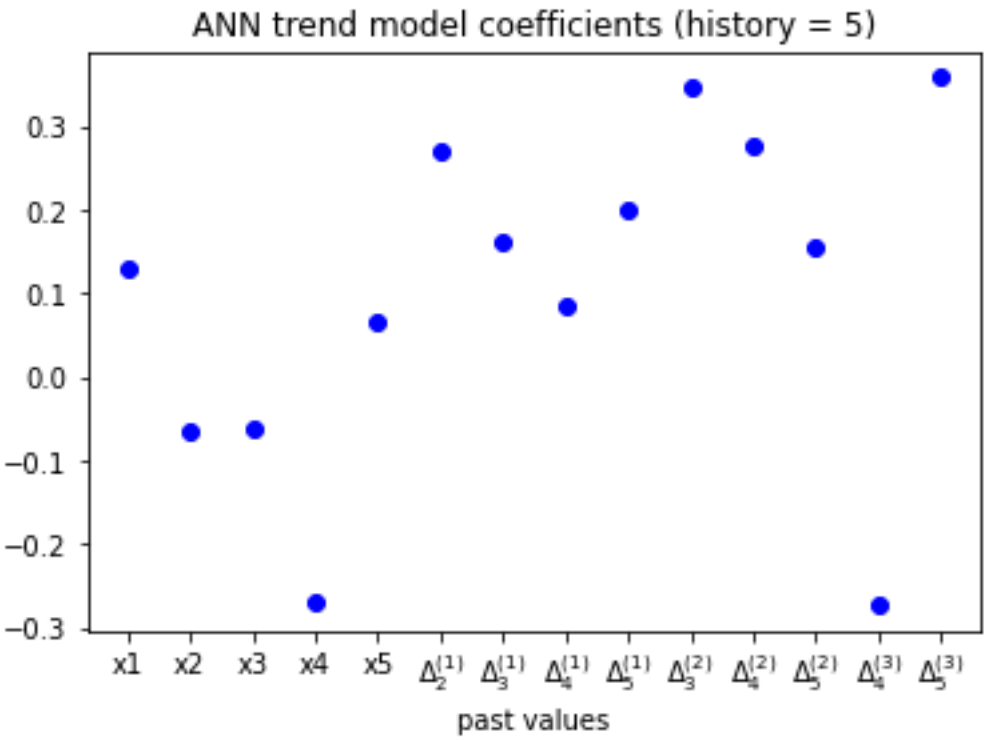

Figure 4.14: ANN trend model coefficients (history = 5)

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusion and future work

We present a close connection between the popular Holt-Winters models and Artificial Neural Network (ANN) based model for time series forecasting. This connection helps understanding how newer, more powerful ANN based models perform relative to traditional models as well as avoiding over-complicating the design of ANN models. Moreover, using such simple ANN models one can easily emulate Holt-Winters models without the burden of searching for best model parameters. Our theoretical arguments and experimental results suggest that ANN models with a single hidden layer and very simple activation function can achieve significantly better results than traditional Holt-Winters models for time series forecasting. Although our current work is limited to the case of time series exponential smoothing and trending, one can easily build on these ideas to tackle seasonality.

For future work we plan to extend our current ideas and explore how Recurrent Neural Network (RNN) [7] based models perform time series forecasting relative to traditional models. RNNs are design to process arbitrary long input sequences (while ANN can process only finite-length input sequences) and can naturally adapt to take into account any number of data points in the past without any design limitation (which is imposed by the input size in ANN models). We believe that such results would help better understanding complex RNN models and possibly simplify their design without sacrificing their performance.

REFERENCES

[1] Medicare Australia, *Monthly anti-diabetic drug sales in australia from 1991 to 2008*, 2024, https://www.key2stats.com/data-sets/listing.

[2] Robert Goodell Brown, *Statistical forecasting for inventory control*, 1960.

[3] Chris Chatfield and Mohammad Yar, *Holt-winters forecasting: Some practical issues*, Journal of the Royal Statistical Society. Series D (The Statistician) **37** (1988), no. 2, 129–140.

[4] François Chollet et al., *Keras*, https://keras.io, 2015.

[5] Inc CiteDrive, *Citedrive brings reference management to overleaf*, 2024, https://otexts.com/fpp3/tspatterns.html [Accessed: (2024)].

[6] _____, *Citedrive brings reference management to overleaf*, 2024, https://otexts.com/fpp3/a-tidy-forecasting-workflow.html [Accessed: (2024)].

[7] Jeffrey L. Elman, *Finding structure in time*, Cognitive Science **14** (1990), no. 2, 179–211.

[8] Tim Hill, Marcus O'Connor, and William Remus, *Neural network models for time series forecasts*, Management Science **42** (1996), no. 7, 1082–1092.

[9] Charles C. Holt, *Forecasting seasonals and trends by exponentially weighted moving averages*, Archives 1982-0001, Carnegie Mellon University, Folder 18, 1957, Tepper School of Business Records.

[10] Rob J Hyndman and George Athanasopoulos, *Forecasting: principles and practice*, OTexts, 2018.

[11] Robin John Hyndman and George Athanasopoulos, *Forecasting: Principles and practice*, 2nd ed., OTexts, Australia, 2018 (English).

[12] UC Irvine, *Machine learning repository*, 2023, Accessed on May 15, 2023.

[13] Prajakta S. Kalekar, *Time series Forecasting using Holt-Winters Exponential Smoothing*, 2004.

[14] Nowrouz Kohzadi, Milton S Boyd, Bahman Kermanshahi, and Iebeling Kaastra, *A comparison of artificial neural network and time series models for forecasting commodity prices*, Neurocomputing **10** (1996), no. 2, 169–181.

[15] Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics **5** (1943), no. 4, 115–133.

[16] Yohana James Mgale, Yunxian Yan, and Shauri Timothy, *A Comparative Study of ARIMA and Holt-Winters Exponential Smoothing Models for Rice Price Forecasting in Tanzania*, OALib (2021).

[17] Fak Mipa, *Forecasting Seasonal Time Series Data using The Holt-Winters Exponential Smoothing Method of Additive Models*, 2020.

[18] J.A. Nelder and R. Mead, *A simplex method for function minimization*, The computer journal **7** (1965), no. 4, 308–313.

[19] Mary Pleños, *Time Series Forecasting Using Holt-Winters Exponential Smoothing: Application to Abaca Fiber Data*, Zeszyty Naukowe SGGW w Warszawie - Problemy Rolnictwa Światowego (2022).

[20] Priyamvada and Rajesh Wadhvani, *Review on various models for time series forecasting*, 2017 International Conference on Inventive Computing and Informatics (ICICI) (2017), 405–410.

[21] Rhuan Carlos Martins Ribeiro, Glauber Tadaiesky Marques, Paulo Cerqueira dos Santos Júnior, J. Felipe Almeida, Pedro Campos, and Otavio Andre Chase, *Holt-Winters Forecasting for Brazilian Natural Gas Production*, International Journal for Innovation Education and Research (2019).

[22] Zaiyong Tang, Chrys De Almeida, and Paul A Fishwick, *Time series forecasting using neural networks vs. box-jenkins methodology*, Simulation **57** (1991), no. 5, 303–310.

[23] Peter R. Winters, *Forecasting sales by exponentially weighted moving averages*, Management Science **6** (1960), no. 3, 324–342.

Appendix A

PYTHON IMPLEMENTATION

```
# -*- coding: utf-8 -*-
"""
author: Kazeem Bankole
"""
#%%***************** imports
    ************************************************
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import keras as keras
from keras.layers import Dense
from keras.utils import set_random_seed

from sklearn.preprocessing import MinMaxScaler
from typing import *

#%%************************** functions
    ****************************************
def ewma(x, alpha: float) -> float:
    if len(x) < 2:
        return x[0]

    return (alpha * x[-1]) + ((1 - alpha) * ewma(x[:-1], alpha))

def holt(x, alpha: float,
            beta: float, initial_trend: float) -> float:
    if len(x) < 2:
        return initial_trend

```

```
28      s = ewma(x, alpha)
29      s0 = ewma(x[:-1], alpha)
30
31      y0 = holt(x[:-1], alpha, beta, initial_trend)
32
33      return (beta * (s - s0)) + ((1 - beta) * y0)
34
35  #%%************************************************ data
        ************************************
36  #data = pd.read_csv('../data/shampoo.csv')["Sales"]
37  data = pd.read_csv('../data/diabetisa10.csv')["Y"][:-125]
38  dataSize = len(data)
39  data = np.array(data.values.tolist())
40
41  future = 2
42  final_times = (dataSize-(future + 1), dataSize-1)
43
44  #%%************************************ experiment1: smoothing
        ************************************
45  alpha = 0.2
46
47  forecast = ewma(data[:-future], alpha=alpha)
48  s = [ewma(data[:i], alpha = alpha) for i in range(1,dataSize-future+1)]
        + ([forecast] * future)
49
50  alpha = 0.75
51
52  forecast2 = ewma(data[:-future], alpha=alpha)
53  s2 = [ewma(data[:i], alpha = alpha) for i in range(1,dataSize-future+1)]
         + ([forecast2] * future)
54
```

```
55
56 #%%****************************************** results
       ****************************************
57 plt.figure(
58     figsize=(16, 10)
59 )
60
61 plt.plot(data, linewidth=3, label='Original data',color="b")
62 plt.plot(s, linewidth=3, label='EWMA (alpha = 0.2)',color="g")
63 plt.plot(s2, linewidth=3, label='EWMA (alpha = 0.75)',color="olive")
64 plt.plot(final_times,
65          [s[-(future + 1)], forecast], linewidth=3,
66          label='Forecast',color="orange")
67 plt.plot(final_times,
68          [s2[-(future + 1)], forecast2], linewidth=3,
69          color="orange")
70 plt.axvspan(*final_times, facecolor='grey', alpha=0.25)
71 plt.legend()
72 plt.show()
73
74 #%%********************************** experiment2: trend
       *****************
75 alpha = 0.2
76 beta = 0.3
77
78 initial_trend = data[1] - data[0]
79 s_estimate = ewma(data[:-(future-1)], alpha)
80 forecast = s_estimate + holt(data[:-(future-1)],
81                                      alpha, beta, initial_trend)
82 s = [data[0]] + [ewma(data[:i], alpha) + holt(data[:i], alpha, beta,
       initial_trend) for i in range(2,dataSize-future+1)]
```

```
83
84 #%%****************************************** results
       ****************************************
85
86 plt.figure(
87     figsize=(16, 10)
88 )
89
90 plt.plot(data, linewidth=3, label='Original data', color = "b")
91 plt.plot(s, linewidth=3, label='Trend', color = "g")
92 plt.plot(final_times,
93          [s[-1], forecast], linewidth=3,
94          label='Forecast', color = "orange")
95 plt.axvspan(*final_times, facecolor='grey', alpha=0.25)
96 plt.legend()
97 plt.show()
98
99 ############################################## ANN based models
       ################################################################
100 #%%************************************* format data
       ****************************************
101 DATACHUNK = 3
102 scaler = MinMaxScaler(feature_range=(0,1))
103
104
105 def ann_smoothing_data(x, datachunk = DATACHUNK, forecast = future):
106     train_data = scaler.fit_transform(np.reshape(x, (-1,1)))
107     # create data arrays
108     x_train = []
109     y_train = []
110     #train data
```

```
    for i in range(datachunk, len(train_data)-forecast):
        x_train.append(train_data[(i-datachunk):i])
        y_train.append(train_data[i:(i+1)])

    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1]))
    y_train = np.reshape(y_train, (y_train.shape[0], y_train.shape[1]))

    return x_train,y_train


def ann_trend_data(x, order = 3, datachunk = DATACHUNK, forecast =
    future):
    dord = min(order, datachunk-1)
    train_data = scaler.fit_transform(np.reshape(x, (-1,1)))
    # create data arrays
    x_train = []
    y_train = []
    #train data
    for i in range(datachunk, len(train_data)-forecast):
        row = []
        row.append(train_data[(i-datachunk):i])
        #add derivatives
        for k in range(dord):
            row.append(train_data[(i-datachunk+k+1):(i)]-train_data[(i-
    datachunk):(i-k-1)])
        x_train.append(np.concatenate(row).ravel())
        y_train.append(train_data[i:(i+1)])

    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1]))
```

```
140     y_train = np.reshape(y_train, (y_train.shape[0], y_train.shape[1]))
141     return x_train,y_train
142 
143 #%%************************************ the ANN model
    ***********************************
144 def identity_activation(x):
145     return x
146 
147 def ann_model(x_train):
148     model = keras.Sequential()
149     model.add(Dense(1, activation=identity_activation, input_shape=(
    x_train.shape[1], )))
150     #model.add(Dense(1, activation="relu", input_shape=(x_train.shape
    [1], )))
151     model.compile(optimizer='adam', loss='mean_squared_error')
152     #model.summary()
153     return model
154 
155 def train(model, x_train, y_train, batch = 4, epochs = 500):
156     history = model.fit(x_train, y_train, batch_size= batch, epochs=
    epochs)
157     return history
158 
159 
160 def ann_forecast(model, x_train, datachunk = DATACHUNK, future = future)
    :
161     x =  np.reshape(x_train[-1], (1, x_train.shape[1]))
162     y = model.predict(x)
163     x = np.concatenate((x[:,1:datachunk],y),axis = 1)
164     f = []
165     f.append(y[0])
```

```
166     for i in range(future):
167         y = model.predict(x)
168         x = np.concatenate((x[:,1:datachunk],y),axis = 1)
169         f.append(y[0])
170     f = np.array(f)
171     #f = f.reshape(-1)
172     return f
173 
174 
175 def ann_trendforecast(model, x_train, datachunk = DATACHUNK, order = 3,
    future = future):
176     dord = min(order, datachunk-1)
177     x =  np.reshape(x_train[-1], (1, x_train.shape[1]))
178     y = model.predict(x)
179     x = np.concatenate((x[:,1:datachunk],y),axis = 1)
180     f = []
181     f.append(y[0])
182     for i in range(future):
183         for k in range(dord):
184             x = np.concatenate((x, x[:,(k+1):(datachunk)]-x[:,(0):(
    datachunk-k-1)]), axis = 1)
185 
186         x =  np.reshape(x, (1, x_train.shape[1]))
187         y = model.predict(x)
188         x = np.concatenate((x[:,1:datachunk],y),axis = 1)
189         f.append(y[0])
190     f = np.array(f)
191     #f = f.reshape(-1)
192     return f
193 
194 
```

```
195
196 #%% ****************************************** train the model:
    smoothing **********************
197 #format data, create model
198 x_train, y_train = ann_smoothing_data(data)
199 model = ann_model(x_train)
200 model.summary()
201
202 # adjust batch size and epochs for accuracy
203 set_random_seed(12345)
204 history = train(model, x_train, y_train)
205
206 # inverse predictions
207 predictions = model.predict(x_train)
208 sann = scaler.inverse_transform(predictions)
209 sx = np.array(range(DATACHUNK, DATACHUNK+len(sann)))
210
211 rmse = np.sqrt(np.mean(sann - scaler.inverse_transform(y_train))**2)
212 print(rmse)
213
214 #predict in the future
215 fut_times = np.array(range(final_times[0], final_times[-1]+1))
216 forecast = scaler.inverse_transform(ann_forecast(model, x_train))
217
218 #%%****************************** plot loss/accuracy
    **********************************
219 plt.figure(
220     figsize=(16, 10)
221 )
222 plt.plot(history.history['loss'])
223 plt.title('ANN smoothing model SSE')
```

```
224 plt.ylabel('SSE')
225 plt.xlabel('epoch')
226 #plt.legend(['piecewise', 'linear'], loc='upper left')
227 plt.show()
228
229
230 #%%**************************** plot the predictions: smoothing
    ************************************************
231 plt.figure(
232     figsize=(16, 10)
233 )
234
235 plt.plot(data, linewidth=3, label='Original data', color = "b")
236 plt.plot(sx, sann, linewidth=3, label='ANN smoothing', color = "g")
237 plt.plot(fut_times,
238          forecast, linewidth=3,
239          label='Forecast', color = "orange")
240 plt.axvspan(*final_times, facecolor='grey', alpha=0.25)
241 plt.title("ANN model (history = " + str(DATACHUNK) + ")")
242 plt.legend()
243 plt.show()
244
245 #%%**************************** plot the weights
    ************************************************
246 z_matrix = model.get_weights()
247
248 TITLE = "ANN smoothing model coefficients (history = " + str(DATACHUNK)
    + ")"
249 sx = []
250 for i in range(DATACHUNK):
251     sx = sx + ['x' + str(i+1)]
```

```
252 plt.figure()
253 plt.plot(sx, z_matrix[0][:,0], 'b',linestyle="",marker="o")
254 plt.title(TITLE)
255 plt.xlabel('past values')
256
257
258
259 ######################################## ANN model with trend
    ########################################
260
261 #%% ****************************************** train the model: trend
    **********************
262 #format data, create model
263 x_train, y_train = ann_trend_data(data)
264 model = ann_model(x_train)
265 model.summary()
266
267 # adjust batch size and epochs for accuracy
268 set_random_seed(12345)
269 history = train(model, x_train, y_train)
270
271 # inverse predictions
272 predictions = model.predict(x_train)
273 sann = scaler.inverse_transform(predictions)
274 sx = np.array(range(DATACHUNK, DATACHUNK+len(sann)))
275
276 rmse = np.sqrt(np.mean(sann - scaler.inverse_transform(y_train))**2)
277 print(rmse)
278
279 #predict in the future
280 fut_times = np.array(range(final_times[0], final_times[-1]+1))
```

```
281 forecast = scaler.inverse_transform(ann_trendforecast(model, x_train))
282
283 #%%********************************* plot loss/accuracy
    ***********************************
284 plt.figure(
285     figsize=(16, 10)
286 )
287 plt.plot(history.history['loss'])
288 plt.title('ANN trend model SSE')
289 plt.ylabel('SSE')
290 plt.xlabel('epoch')
291 #plt.legend(['piecewise', 'linear'], loc='upper left')
292 plt.show()
293
294
295 #%%*************************** plot the predictions: smoothing
    ************************************************
296 plt.figure(
297     figsize=(16, 10)
298 )
299
300 plt.plot(data, linewidth=3, label='Original data', color = "b")
301 plt.plot(sx, sann, linewidth=3, label='ANN trend model', color = "g")
302 plt.plot(fut_times,
303          forecast, linewidth=3,
304          label='Forecast', color = "orange")
305 plt.axvspan(*final_times, facecolor='grey', alpha=0.25)
306 plt.title("ANN model (history = " + str(DATACHUNK) + ")")
307 plt.legend()
308 plt.show()
309
```

```
#%%**************************** plot the weights
    ************************************************
z_matrix = model.get_weights()

TITLE = "ANN trend model coefficients (history = " + str(DATACHUNK) + ")
    "
#sx = np.array(range(x_train.shape[1])) + 1
sx = []
for i in range(DATACHUNK):
    sx = sx + ['x' + str(i+1)]

dord = min(3, DATACHUNK-1)
for o in range(dord):
    for i in range(DATACHUNK-o-1):
        sx = sx + ['$\\Delta^{(' + str(o+1) + ')}_{' + str(i+1 + o + 1)
    + '}$']

plt.figure()
plt.plot(sx, z_matrix[0][:,0][::-1], 'b',linestyle="",marker="o")
plt.title(TITLE)
plt.xlabel('past values')
```

Listing A.1: The complete Python code for models and experiments